

## Games vs. search problems

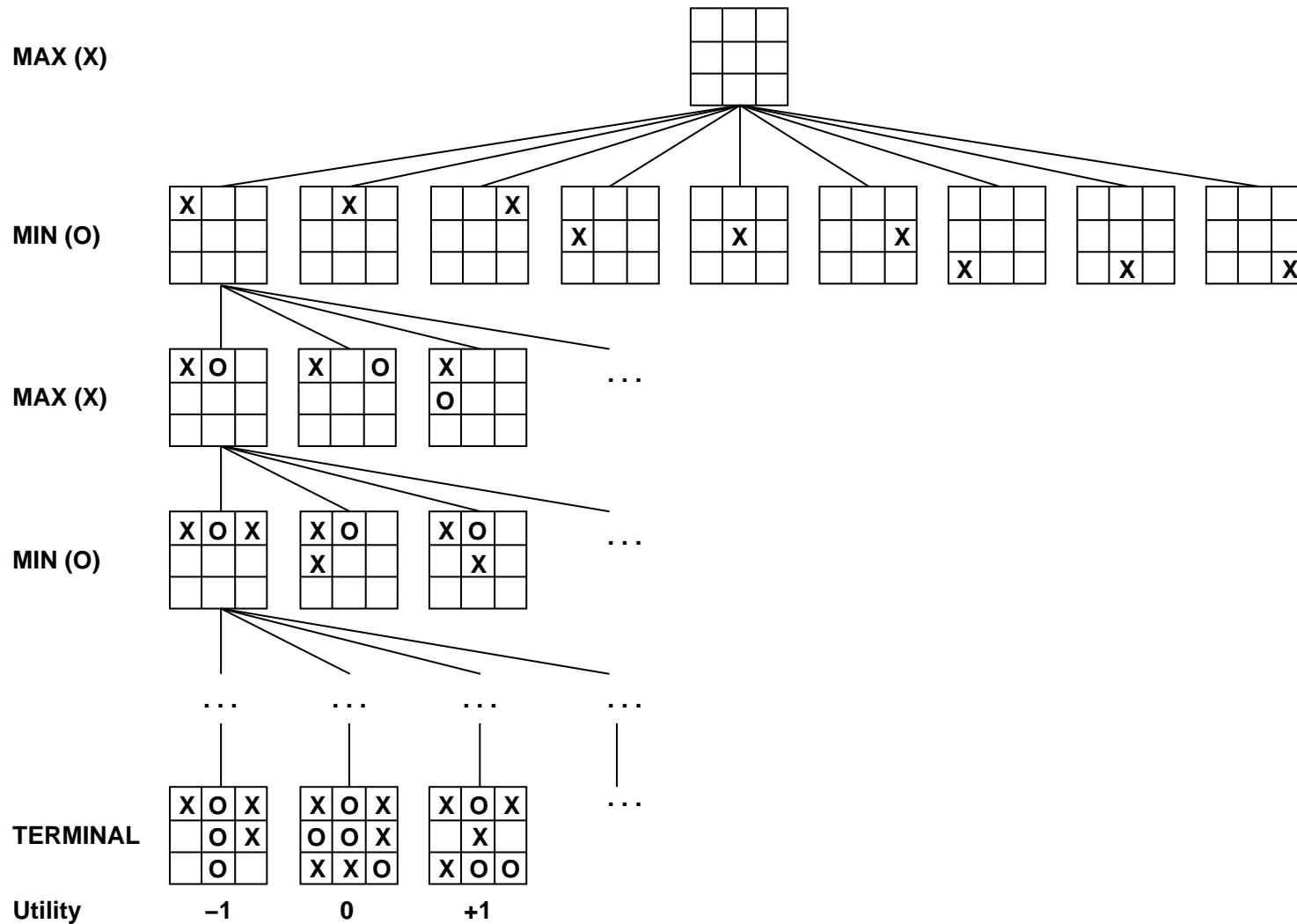
“Unpredictable” opponent  $\Rightarrow$  solution is a **strategy**  
specifying a move for every possible opponent reply

Time limits  $\Rightarrow$  unlikely to find goal, must approximate

Plan of attack:

- Computer considers possible lines of play (Babbage, 1846)
- Algorithm for perfect play (Zermelo, 1912; Von Neumann, 1944)
- Finite horizon, approximate evaluation (Zuse, 1945; Wiener, 1948; Shannon, 1950)
- First chess program (Turing, 1951)
- Machine learning to improve evaluation accuracy (Samuel, 1952–57)
- Pruning to allow deeper search (McCarthy, 1956)

# Game tree (2-player, deterministic, turns)

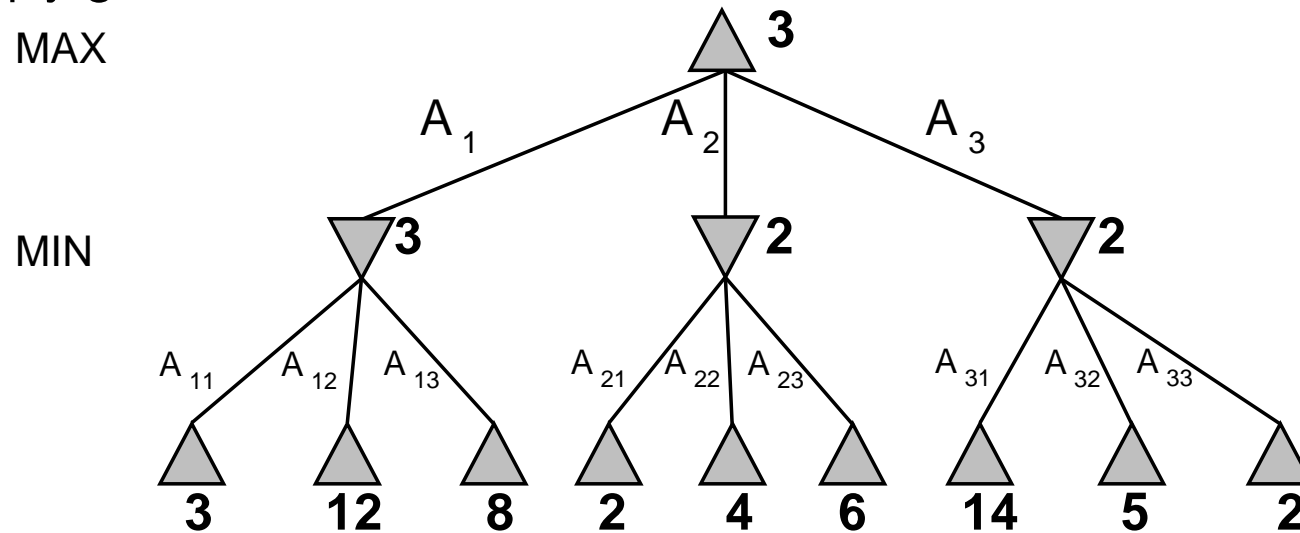


# Minimax

Perfect play for deterministic, perfect-information games

Idea: choose move to position with highest **minimax value**  
= best achievable payoff against best play

E.g., 2-ply game:



# Minimax algorithm

**function** MINIMAX-DECISION(*state*) **returns** *an action*

**inputs:** *state*, current state in game

**return** the *a* in ACTIONS(*state*) maximizing MIN-VALUE(RESULT(*a*, *state*))

---

**function** MAX-VALUE(*state*) **returns** *a utility value*

**if** TERMINAL-TEST(*state*) **then return** UTILITY(*state*)

$v \leftarrow -\infty$

**for** *a, s* in SUCCESSORS(*state*) **do**  $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(s))$

**return** *v*

---

**function** MIN-VALUE(*state*) **returns** *a utility value*

**if** TERMINAL-TEST(*state*) **then return** UTILITY(*state*)

$v \leftarrow \infty$

**for** *a, s* in SUCCESSORS(*state*) **do**  $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(s))$

**return** *v*

## Properties of minimax

Complete?? Yes, if tree is finite (chess has specific rules for this)

Optimal?? Yes, against an optimal opponent. Otherwise??

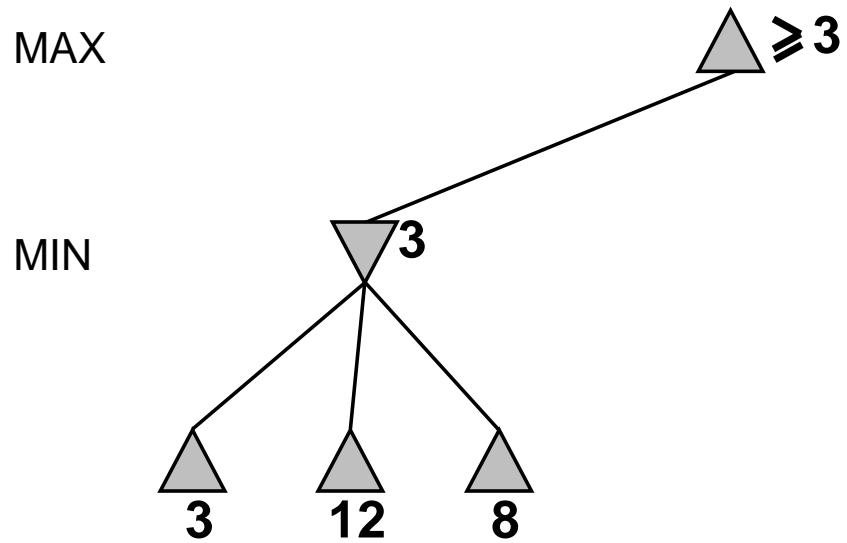
Time complexity??  $O(b^m)$

Space complexity??  $O(bm)$  (depth-first exploration)

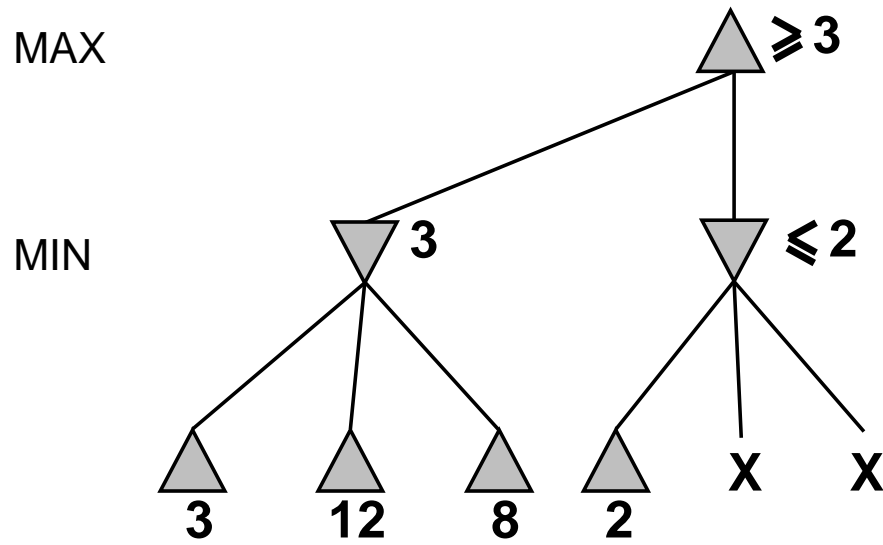
For chess,  $b \approx 35$ ,  $m \approx 100$  for “reasonable” games  
 $\Rightarrow$  exact solution completely infeasible

But do we need to explore every path?

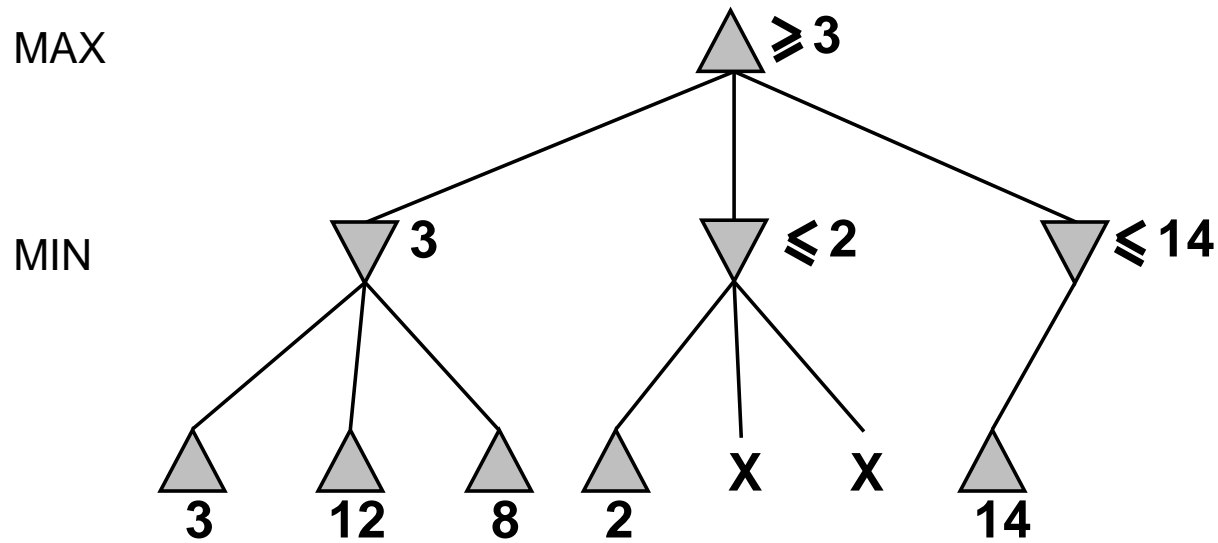
## $\alpha$ - $\beta$ pruning example



## $\alpha$ - $\beta$ pruning example

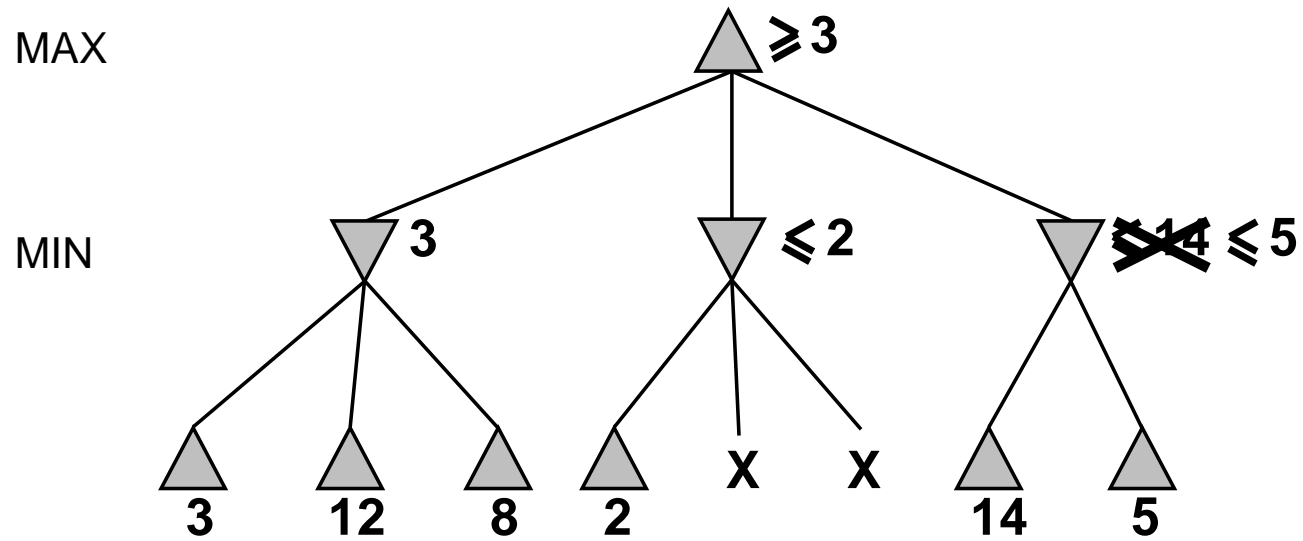


## $\alpha$ - $\beta$ pruning example

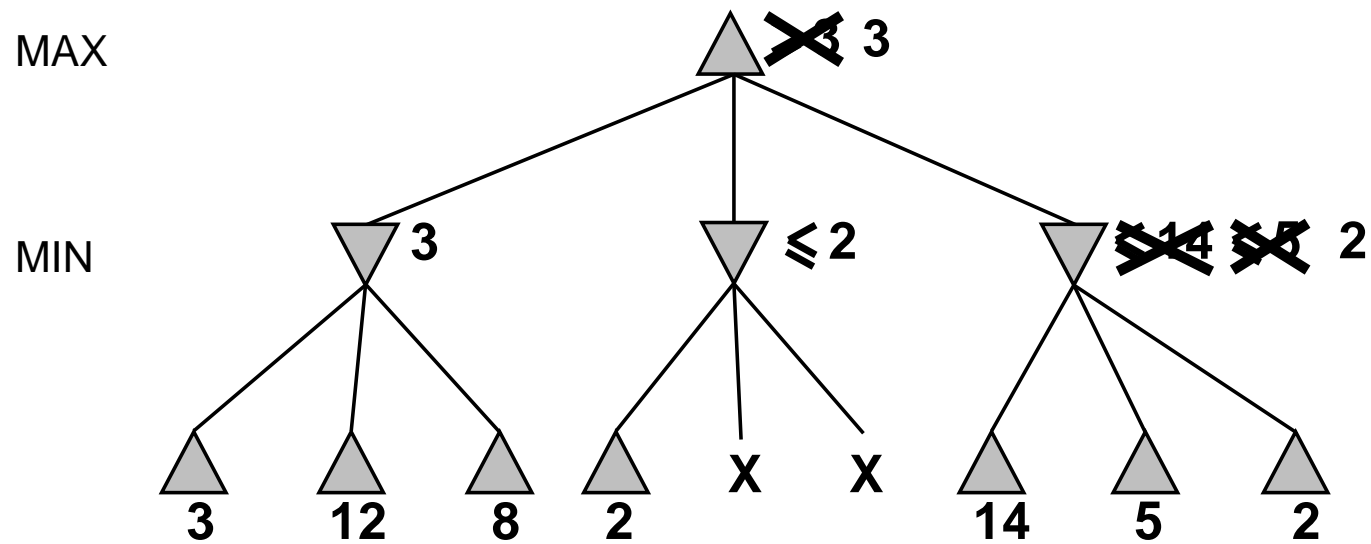




# $\alpha$ - $\beta$ pruning example



# $\alpha$ - $\beta$ pruning example



## The $\alpha$ - $\beta$ algorithm

**function** ALPHA-BETA-DECISION( $state$ ) **returns** an action  
    **return** the  $a$  in ACTIONS( $state$ ) maximizing MIN-VALUE(RESULT( $a$ ,  $state$ ))

---

**function** MAX-VALUE( $state, \alpha, \beta$ ) **returns** a utility value  
    **inputs:**  $state$ , current state in game  
         $\alpha$ , the value of the best alternative for MAX along the path to  $state$   
         $\beta$ , the value of the best alternative for MIN along the path to  $state$   
    **if** TERMINAL-TEST( $state$ ) **then return** UTILITY( $state$ )  
     $v \leftarrow -\infty$   
    **for**  $a, s$  in SUCCESSORS( $state$ ) **do**  
         $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(s, \alpha, \beta))$   
        **if**  $v \geq \beta$  **then return**  $v$   
         $\alpha \leftarrow \text{MAX}(\alpha, v)$   
    **return**  $v$

---

**function** MIN-VALUE( $state, \alpha, \beta$ ) **returns** a utility value  
    same as MAX-VALUE but with roles of  $\alpha, \beta$  reversed

## Properties of $\alpha-\beta$

Pruning **does not** affect final result

Good move ordering improves effectiveness of pruning

With “perfect ordering,” time complexity =  $O(b^{m/2})$   
 $\Rightarrow$  **doubles** solvable depth

A simple example of the value of reasoning about which computations are relevant (a form of **metareasoning**)

Unfortunately,  $35^{50}$  is still impossible!

## Resource limits

Standard approach:

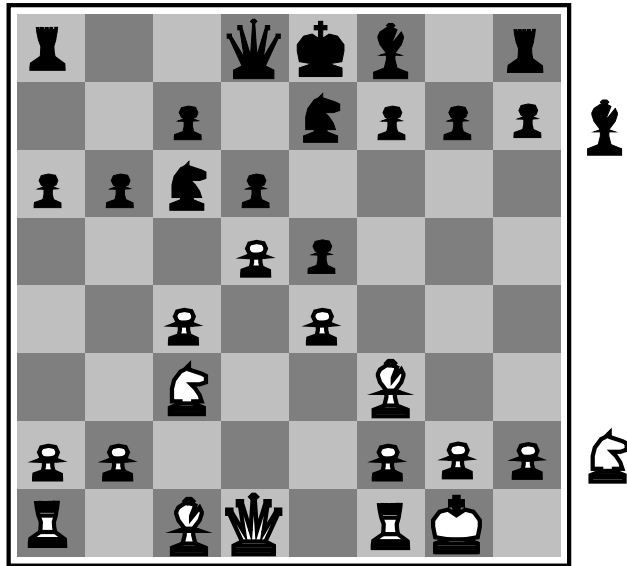
- Use CUTOFF-TEST instead of TERMINAL-TEST  
e.g., depth limit (perhaps add quiescence search)
- Use EVAL instead of UTILITY  
i.e., evaluation function that estimates desirability of position

Suppose we have 100 seconds, explore  $10^4$  nodes/second

$\Rightarrow 10^6$  nodes per move  $\approx 35^{8/2}$

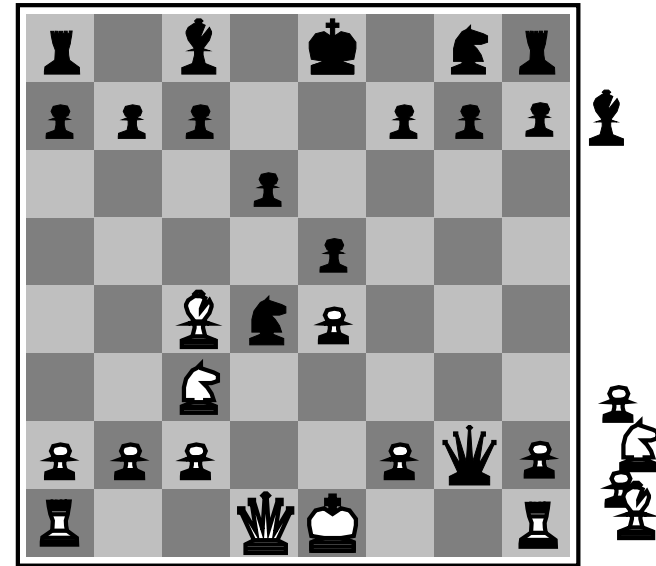
$\Rightarrow \alpha\text{-}\beta$  reaches depth 8  $\Rightarrow$  pretty good chess program

# Evaluation functions



**Black to move**

**White slightly better**



**White to move**

**Black winning**

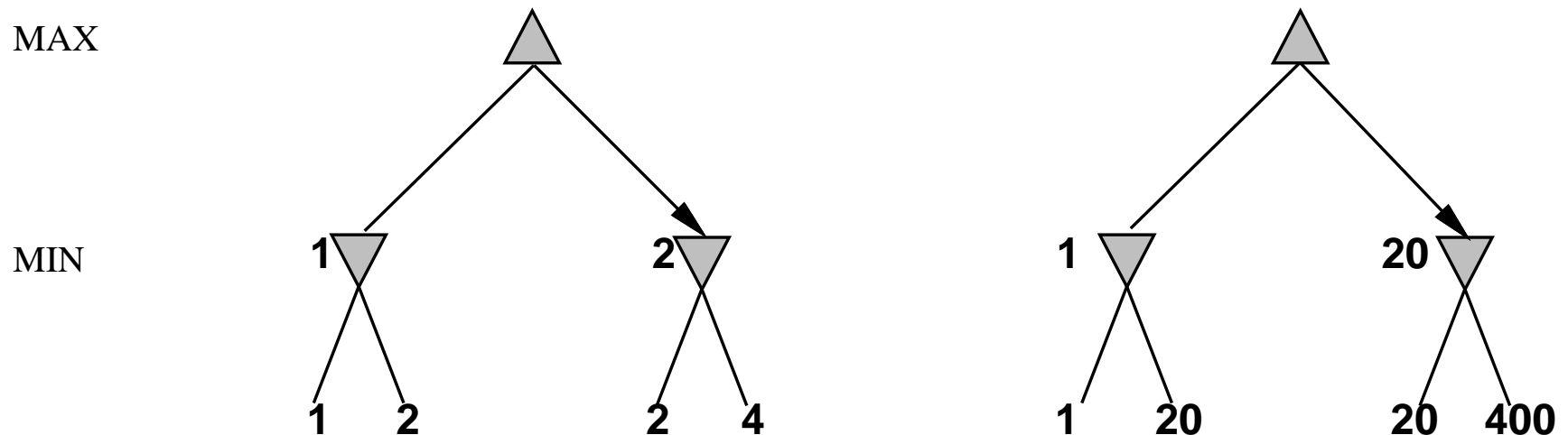
For chess, typically **linear** weighted sum of **features**

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \dots + w_n f_n(s)$$

e.g.,  $w_1 = 9$  with

$f_1(s) = (\text{number of white queens}) - (\text{number of black queens}), \text{ etc.}$

## Digression: Exact values don't matter



Behaviour is preserved under any **monotonic** transformation of  $\text{EVAL}$

Only the order matters:

payoff in deterministic games acts as an **ordinal utility** function